

An Adaptive, High Performance MAC for Long-Distance Multihop Wireless Networks

Sergiu Nedeveschi[†] Rabin Patra[†] Sonesh Surana[†] Sylvia Ratnasamy*
Lakshminarayanan Subramanian[‡] Eric Brewer^{†*}

[†] University of California, Berkeley
{sergiu,rkpatra,sonesh,brewer}@cs.berkeley.edu

* Intel Research, Berkeley [‡] New York University
sylvia.p.ratnasamy@intel.com lakshmi@cs.nyu.edu

ABSTRACT

We consider the problem of efficient MAC design for long-distance WiFi-based mesh networks. In such networks it is common to see long propagation delays, the use of directional antennas, and the presence of inter-link interference. Prior work has shown that these characteristics make traditional CSMA-based MACs a poor choice for long-distance mesh networks, prompting several recent research efforts exploring the use of TDMA-based approaches to media access. In this paper we first identify, and then address, several shortcomings of current TDMA-based proposals, which exhibit inefficient throughput and delay characteristics as they use fixed-length transmission slots that cannot adapt to dynamic variations in traffic load. We show that throughput achieved by existing solutions falls far short of the optimal achievable network throughput. Current TDMA-based solutions also only apply to bipartite network topologies due to the scheduling constraints imposed by inter-link interference.

In this paper, we present JazzyMac, a simple, practical and efficient MAC protocol that addresses the above limitations. JazzyMac achieves efficiency by enabling variable-length link transmissions slots; each node can adapt the length of their transmission slots in accordance with changing traffic demands. JazzyMac is practical as it can be applied to arbitrary network topologies, and each node can use purely local information for slot adaptation. Finally, the use of dynamic slot lengths allows JazzyMac to achieve better tradeoffs between throughput and delay.

We evaluate JazzyMac using detailed simulation over a range of traffic patterns and realistic topologies. Our results show that JazzyMac improves throughput in *all* considered scenarios. This improvement is often substantial (*e.g.*, in 50% of our scenarios, throughput improves by over 40%) and is particularly pronounced for the common case of asymmetric traffic (*e.g.*, leading to almost 100% improvements). Furthermore, compared to current solutions, JazzyMac can achieve much better average delay for the same throughput.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MobiCom'08, September 14–19, 2008, San Francisco, California, USA.
Copyright 2008 ACM 978-1-60558-096-8/08/09 ...\$5.00.

Categories and Subject Descriptors

C.2.1 [Computer Systems Organization]: Computer-Communication Networks—*Network Architecture and Design*

General Terms

Algorithms, Design, Performance

Keywords

Multi-hop wireless, Long-distance point-to-point wireless, MAC protocols

1. INTRODUCTION

Multi-hop WiFi long-distance networks (WiLD) networks have become increasingly popular in the last few years, providing cost-efficient connectivity to sparsely populated areas and rural regions in developing and industrialized countries alike. Example deployments include the Digital Gangetic Plains project [19], the AirJaldi and Aravind networks [23] and the Akshaya network [24]. These networks serve thousands of users, providing videoconferencing and VoIP services in addition to basic Internet access.

Due to the presence of long-distance links and their use of directional antennas, WiLD networks present unique challenges relative to traditional short-range mesh networks [3]. Specifically, these networks suffer from long propagation delays and an increased likelihood of inter-packet collisions. In addition, deployments where only a small number of non-overlapping wireless channels are available also suffer from inter-link interference. Prior work has shown that these challenges make traditional MACs based on carrier sensing, a poor fit for WiLD networks [21]. To address these challenges, several TDMA-based MAC solutions such as 2P [19] and WiLDNet [16] have been developed and are currently used in practical deployments. This paper identifies and addresses certain key performance limitations in 2P and WiLDNet. These limitations arise primarily because these solutions rely on a TDMA schedule with fixed-length slots and hence cannot adapt to dynamic traffic variations.

In this paper, we first gauge the potential for improved performance that might result from a MAC solution that takes advantage of observed traffic conditions. For this, we compute the optimal throughput achievable in WiLD networks assuming complete knowledge of the network traffic workload (our computation here extends prior work [11, 14]). Our results reveal that current static approaches leave much room for improvement. Spurred on by this finding, we then design JazzyMac, a simple, practical and efficient MAC protocol for WiLD networks. The key innovation in JazzyMac is dynamic slot adaptation. Using JazzyMac, nodes adapt the

length of their transmission slots to changing traffic conditions. JazzyMac’s dynamic slot adaptation (compared to the fixed-length slots of prior work) enables more efficient use of network capacity by adapting to traffic and by allowing more parallel communication among non-interfering links. Dynamic slot sizes also enable control over the bandwidth-delay tradeoff. JazzyMac is simple and practical. The protocol is fully distributed allowing each node to adapt its transmission slot size using purely local state and locally-observed traffic information. We also show that JazzyMac’s distributed protocol is provably deadlock free.

In summary, JazzyMac offers three key advantages relative to prior work:

1. Performance: JazzyMac achieves superior throughput (with up to 100% improvement) over 2P and WiLDNet across *all* network sizes, topologies, and traffic workloads. Moreover, this improvement increases dramatically in the case of asymmetric traffic, a commonly-occurring workload in rural Internet access.

2. Controlled throughput-vs-delay tradeoff: JazzyMac offers network operators flexibility in navigating the throughput-vs-delay tradeoff achieved by the network. This allows JazzyMac to better support applications such as Internet telephony that favor low delays over higher throughput. Moreover, compared to existing protocols, we show that JazzyMac achieves consistently lower latency even when achieving higher throughputs.

3. Support for topologies beyond bipartite graphs: When using only a single channel, current solutions require that the topology over which they operate be bipartite; for non-bipartite topologies the solutions are applied only to a bipartite subgraph of the overall topology graph. JazzyMac eliminates this constraining requirement and achieves better performance (and fault tolerance) using the complete graph rather than just a bipartite subgraph. Eliminating this requirement is important because it allows network deployments to grow at will, without any topology-related constraints (beyond the usual line-of-sight).

The remainder of this paper is organized as follows: we start in Section 2 with an overview of long-distance multi-hop wireless networks, then examine several opportunities for improvement (Section 3). We describe the design of JazzyMac in Section 4 and evaluate its performance in Section 5. We discuss related work in Section 6 and finally conclude in Section 7.

2. BACKGROUND

WiLD networks are multi-hop wireless networks, featuring long point-to-point wireless links that can range from tens to hundreds of kilometers. Some real-life deployments of WiLD networks include the Akshaya network [24], the Digital Gangetic Plains project [2], the CRCnet project [6], and our own project, the Aravind network [23]. Recently, we have also successfully deployed several links exceeding 100 km, including a 382 km link in Venezuela [23].

In these networks, wireless nodes are equipped with multiple radios, co-located on the same tower. To achieve long range, WiLD deployments use high-power 802.11 radios (400mW), and each radio is connected to a directional antenna, with gains as high as 30dBi. These radios can operate on the same wireless channel, or on different (non-overlapping) channels. Although operation on different channels avoids inter-radio interference, there are several practical constraints that may require the operation of co-located radios on the same channel. This can happen due to country-specific regulatory restrictions [22], high spectrum costs, or the limited number of available channels. For example, 802.11b has only 3 non-overlapping channels, which forces nodes with many radios

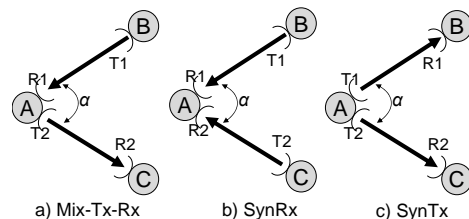


Figure 1: SynOp scenarios for interference

to reuse some of the channels.

We therefore focus our study on single-channel operation. Our results however still apply when multiple non-overlapping channels are available. In such cases, the network can be partitioned into multiple independent components, each operating on a different channel; our solution then applies to finding how to schedule these sub-components. In this respect, our work is orthogonal to that by Raman *et al.* [17], exploring optimal solutions to partitioning the network into single-channel subgraphs, given multiple available channels.

2.1 Using SynOp to avoid interference

Long-distance links with high-gain directional antennas interfere with each other in a very specific manner. More precisely, as first observed by Raman *et al.* [18], and reiterated in [19, 16], co-located radios (same physical location) operating on the same wireless channel interfere with each other if one of them transmits while the other receives. However, two adjacent directional links that either transmit simultaneously (*SynTx*), or receive simultaneously (*SynRx*), will be largely *interference-free* – a mode of operation termed as Simultaneous Synchronized Operation (*SynOp*).

We briefly explain the reason for this behavior. Consider the adjacent directional point-to-point links depicted in Figure 1, separated by an angle α . Now consider the following three potential interference scenarios:

Mix-Tx-Rx: In this scenario, depicted in Figure 1(a), T_2 ’s transmissions interfere with R_1 ’s reception, due to the physical proximity between the radios and the presence of antenna side-lobes. Therefore, operating the links in this mode is not feasible.

SynRx: During simultaneous receive, shown in Figure 1(b), T_2 ’s transmissions are seen as interference at R_1 , and T_1 ’s transmissions are seen as interference at R_2 . For the interfering signal to be ignored, the difference between useful signal and interference must be larger than a certain threshold $Th_{isolation}$, which depends on modulation and data-rate; *e.g.*, with 802.11b at 11Mbps, $Th_{isolation} \approx 10dB$ [21, 18]. Fortunately, this level of isolation can usually be ensured through the difference in gain levels provided by the directional antennas, if the links are separated by a sufficiently large angle. If we denote the difference between the antenna gain of the main lobe and the gain at an angle α away from the main lobe by S_{alpha} (also called the rejection level at angle α), then adjacent links are interference free under the following condition [18]:

$$|P_{R1} - P_{R2}| < S_{\alpha} - Th_{isolation} \quad (1)$$

where P_{R1} and P_{R2} are the receive power levels at R_1 and R_2 respectively.

For example, if links use typical 24dBi grid antennas [10] (also used in our deployments) in horizontal polarization, an angular separation of more than 10° (half the width of the antenna main lobe) translates into an isolation of at least 25dB (sometimes larger, not monotonically increasing with the separation angle). This means that 802.11b links receiving simultaneously are interference-free if $|P_{R1} - P_{R2}| < 15dB$. This can be easily satisfied by a large range

of values (e.g., $P_{R1} = P_{R2}$), and even if the path loss of the two links is very different, the condition can be satisfied by adjusting the radio transmit power accordingly (by reducing the TX power on the stronger link).

SynTx: With simultaneous transmissions, as in Figure 1(c), interference may occur at nodes B and C , but not at node A . Once again, R_1 may see interference from T_2 , and R_2 from T_1 . Given the symmetry of the two links, ensuring non-interference during SynTx can be done by enforcing a similar condition to that in equation 1.

We note that simultaneous transmission is infeasible using a carrier-sensing MAC, such as 802.11, since radios can hear each others transmission, causing one of the radios to backoff. However this is not an issue with MACs such as 2P, WildNet and this paper’s JazzyMac since they do not rely on carrier sensing.

In summary, simultaneous synchronized operation (*SynOp*) can allow multiple adjacent WiLD links to simultaneously use the same wireless channel *provided* the links are separated by a sufficiently large angle α and the radio transmit powers are chosen to satisfy the constraint from equation 1. Given the gain pattern of typical grid directional antennas [10], an angular separation α larger than 30° provides generous isolation between adjacent links; this has also been demonstrated experimentally [18, 19] and validated in our deployments [16, 23]. This separation limits the connectivity degree to at most 12 adjacent links on the same channel, a number higher than that reported by any existing deployments. Moreover, having adjacent links with an angular separation smaller than the threshold is also possible through the use of cross-polarization in which the antennas of one link use vertical polarization, and the antennas of the other link use horizontal polarization. For the antennas in our network deployments, this adds an extra $26dB$ isolation [10]. We therefore assume that synchronized simultaneous operation is feasible between any two adjacent links, and use this assumption for the remainder of this paper.

2.2 MAC protocols for WiLD links

CSMA-based MAC protocols have been shown to perform poorly in networks with long distance links [19, 21], leading to a preference for TDMA-based MAC solutions. 2P [19] was the first to propose a TDMA-based approach for WiLD networks; WiLDNet [16] extended the 2P approach with techniques to deal with packet loss and to improve end-to-end performance in multi-hop long-distance networks.

In these MACs, long-distance links alternate between *transmit* and *receive* slots of fixed lengths. Inter-link interference is avoided by eliminating the situation in which a node transmits on one link while receiving on another. Therefore, wireless nodes can either send on some of their links, or receive on some of their links, but not both. These constraints can be efficiently met in bipartite network topologies, as they allow nodes to use all of their links simultaneously and alternate as a group between send mode and receive mode. 2P and WiLDNet are thus designed to work in bipartite topologies.

Figure 2(a) shows an example of such a bipartite network. Using 2P or WiLDNet, all nodes in partition A first transmit on all of their links (for a time slot of size $t_{A \rightarrow B}$). Following this, all nodes in partition B transmit on all their links (for a time slot of $t_{B \rightarrow A}$). The ratio between these slot sizes regulates the bandwidth allocation for every network link between the two partitions. In practice, $t_{A \rightarrow B}$ and $t_{B \rightarrow A}$ are almost always set to be equal since this maximizes throughput for traffic paths spanning more than two hops [17, 19]. Given the significant similarities between 2P and WiLDNet, we henceforth refer to both collectively as Fixed TDMA (FT).

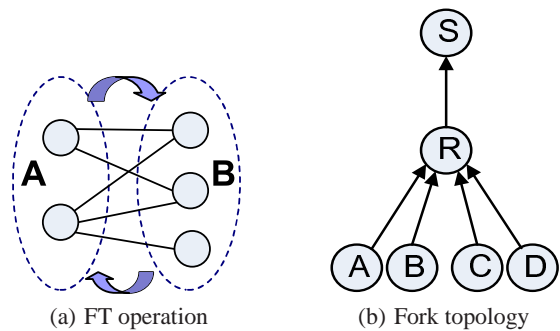


Figure 2: Example topologies

3. OPPORTUNITIES FOR IMPROVEMENT

The 2P and WiLDNet solutions described in the previous section represent important theoretical and practical advances. They successfully cope with the problems due to long propagation delays and inter-link interference and have been successfully deployed in numerous networks [4, 23], serving many thousands of users. Nonetheless, we believe there is significant, as-yet untapped, potential to further improve network performance; specifically, to increase network throughput and reduce latency. Additionally, we believe there is room to improve spectrum usage by making better use of a single channel. In this section we explore these opportunities qualitatively and then, in the following section, quantify the potential for improved network utilization by comparing the throughputs achieved by current solutions to upper bounds computed by optimal offline algorithms.

3.1 Improving Throughput

We discuss two important avenues that can significantly improve the throughput achieved in WiLD networks.

1) Adapting to Traffic Demand: Current MAC solutions for WiLD networks feature a static TDMA slot allocation. This approach is simple, robust, and easy to deploy. However we conjecture that higher throughputs could be achieved by having nodes exploit information about current traffic demands to dynamically adapt their transmission slot sizes. The following examples illustrate this intuition:

Example 1: Single link: Consider the simplest case of a network with a single link between nodes A and B and assume that the traffic demand only exists from A to B . In this scenario, the highest throughput would be achieved by configuring the link to transmit from A to B for (almost) the entire time. This can be achieved by allocating large transmit slots in the direction $A \rightarrow B$, and very short transmit slots in the reverse direction. If subsequently the direction of traffic flow is reversed, then the optimal slot allocation would correspondingly change, with longer slots from B to A . If we were to use such an adaptive approach, the unidirectional traffic could always be served at close to the full link capacity. Unfortunately, approaches with fixed slot sizes cannot deliver similarly high throughputs. Instead, in these approaches, the link is always scheduled to transmit for $x\%$ of the time in direction $A \rightarrow B$ and $1 - x\%$ in the reverse direction, with a typical setting of $x = 50\%$.

Example 2: A fork topology: Figure 2(b) illustrates yet another example. In this scenario, we have a sink node S , and several source nodes A, B, C , and D connected to the sink through relay node R . Let us assume all links have the same data rate, and analyze the optimal slot size allocation for relay R . If only one of the sources (say A) sends traffic to the sink, the slot allocation that maximizes

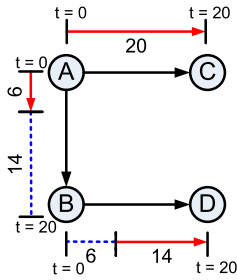


Figure 3: Overlap of transmissions

throughput is the one in which node R has equally sized transmit and receive slots. In this case, R receives data for 50% of the time, and relays this data for the remainder 50% of the time. Now assume that we have 2 sources sending to S . In this case, the bandwidth-optimal solution would be to have R receive for $1/3$ of the time (from both senders), and then relay this data to S in the remaining $2/3$ of the time. Thus, R would have a transmit slot twice as long as the receive slot. Similarly, if all four sources are sending traffic, the best scenario would be the one in which the transmit slot at R is 4 times longer than its receive slot.

In each of the previous examples, a simple strategy to take advantage of local traffic information is to monitor the volume of traffic on outgoing links and then adapt the size of TDMA transmit slots to be proportional to the volume of traffic to be transmitted. This is the fundamental intuition behind JazzyMac.

2) Allowing neighboring transmissions that overlap: Current MAC protocols such as 2P and WiLDNet require that a node maintain all of its links in transmit mode for the same (fixed) time duration. However, there are several situations where this can be needlessly inefficient. For example, consider the topology presented in Figure 3, in which traffic flows are represented by arrows. In this topology, since nodes A and B are neighbors, they can never simultaneously operate in transmit mode (as per current protocols). However, it is possible that the traffic demand is such that A only needs a portion of its transmit slot to B (from say, $t = 0$ to $t = 6$). In this case, we can allow B to start transmitting to a third node (D) at an earlier time ($t = 6$) rather than having to wait until the end of A 's transmission slot ($t = 20$). This means that, for a portion of their transmission slots, both A and B can transmit simultaneously while still respecting all the invariants required to avoid interference. Such *neighboring-but-independent* transmissions have the potential to further increase network channel utilization and our JazzyMac protocol is designed to exploit these opportunities.

3.2 Improving the bandwidth-delay tradeoff

Besides network throughput, another issue of particular interest in long-distance networks is the per-packet delay. Although a large fraction of the popular applications over WiLD networks are delay-sensitive such as telemedicine [23] and VoIP [24], existing solutions introduce significant per-hop delays.

One of the main reasons for larger delays is the TDMA approach adopted by current protocols, and the fact that practical constraints prevent TDMA slot sizes from being very small. This happens because switching between a *sending slot* and a *receiving slot* cannot be done instantaneously; it requires a non-zero *guard time* in which packets are neither transmitted nor received [16].

A lower bound for the size of this guard time is the round-trip propagation delay, which is significant in long-distance networks.

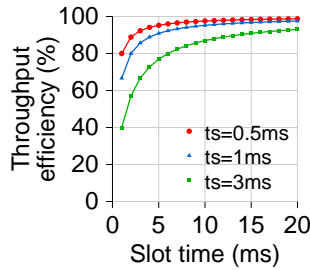


Figure 4: Throughput efficiency vs. slot time

For example, a $75km$ link has a round-trip delay of $0.5ms$. Also, in order to maintain synchronization in the network, the size of the guard time is constrained by the round-trip delay of the *longest* link in the network [19].

Besides propagation delay, existing implementations feature additional constraints that make this guard time much larger in practice. This is especially true of implementations on top of WiFi hardware, because the TDMA mechanisms are not supported in the PHY layer (and firmware), but implemented either in the WiFi driver or above it. This introduces additional (sometimes variable) delays between the time a packet is sent from the driver and the actual time that the packet is sent over the air. Because of these inefficiencies, the guard time in WiLDNet is $3ms$.

Having a large slot guard time t_{switch} limits the minimum slot size. This in turn affects the average per-hop delay, which is proportional to the slot time. For example, the average delay when very lightly utilized is $(t_{switch} + t_{slot})^2 / 2(2t_{slot} + t_{switch}) \approx t_{slot}/4$, while the maximum per-hop delay at close to saturation utilization is $\approx 2t_{slot}$. Figure 4 plots the bandwidth as a function of slot time, assuming guard times t_{switch} of $0.5ms$, $1ms$ and $3ms$.

Since existing approaches use fixed slots, the bandwidth vs. delay tradeoff is fixed, usually to a value that favors bandwidth while sacrificing delay (e.g., a $10ms$ slot). In small deployments this is acceptable, but with larger-scale networks the average hop count increases, the end-to-end delay penalty becomes prohibitive for interactive applications.

We believe that dynamic slot adaptation can alleviate this problem. This would allow for the bandwidth-delay tradeoff to be negotiated differently for different links, taking into account traffic demand. Links seeing low utilizations could utilize small TDMA slots and deliver low per-hop delay, since maximum link bandwidth would not be necessary to serve the traffic demand. Conversely, for highly utilized links the tradeoff could be shifted towards supporting higher maximum bandwidth, by using larger slots (e.g., $20ms$). This approach would allow the network to achieve the best of both worlds: small average delays and maximum bandwidth efficiency when required.

3.3 Single-channel operation on arbitrary topologies

Sending simultaneously and receiving simultaneously on all of a node's links avoids link interference, is very simple to operate, and easy to implement. It is also a very efficient way to operate if the network topology happens to be bipartite, and existing approaches (2P and WiLDNet) take advantage of this.

Unfortunately, enforcing the network topology to be bipartite can be limiting, because it constrains the ways in which networks can be gradually extended. For example, consider the case when a new network node A is added to the network, and A has line of sight to nodes B and C . If B and C are already connected to each other, node A can only connect to one of the two (in order to maintain the bipartite constraint). For node A this implies that a) it cannot have redundant links, making network connectivity less reliable, and b) it is served at suboptimal network capacity.

Raman [17] proposes a solution to address this when several non-overlapping channels are available, by dividing the network into bipartite subgraphs operating on different channels, and using 2P on each of these subgraphs.

However, under the constraint of single-channel operation, such an approach cannot be used. We therefore investigate the following intuitive ways to adapt the TDMA scheme in which nodes send or receive on all of their links for use in non-bipartite topologies:

1. *FT*: Fixed-slot TDMA according to vertex colors. First compute the minimum vertex coloring of the graph. Then nodes transmit in TDMA slots, according to their color. Colors are scheduled for transmission in a round-robin fashion, and therefore each node sends once every K slots, where K is the number of colors. For bipartite graphs (which can be colored with 2 colors), the behaviour of this algorithm is the equivalent to the that of 2P and WiLDNet. (We will describe a slightly more efficient version of this approach in section 4).
2. *FT-CUT*: Fixed-slot TDMA over maxcut. We first compute the maximal subgraph that is bipartite and contains all the network nodes — i.e. a *maxcut* in the original graph. We then use 2P on the maxcut, keeping other links as backups.

The latter approach features two types of links: some that are used for the entire time (to either send or receive), and others that are never used in normal operation. The former uses all the links, but all of them are only used for part of the time ($2/K$ of the time). Dynamic slot sizes can work with either approach and we compare the efficiency of these approaches, with and without adaptive slot sizes, later in the paper.

3.4 Quantifying the Throughput Gap

The shortcomings described in the previous section point to the fact that existing solutions are likely to yield suboptimal throughput. In order to measure how far are these approaches from being optimal, we investigate ways to compute a link transmission schedule that optimizes total network throughput. This bound can then be used to quantify the inefficiency of practical protocols.

Throughput-Optimal Link Schedule: We borrow from prior work [11] in the more general context of multi-hop wireless networks featuring inter-link interference. In this work, optimal link scheduling is framed as a *max-flow optimization problem*, with an additional constraint that avoids inter-link interference by enforcing that interfering links never schedule transmissions simultaneously.

Interference in a connectivity graph G can be specified by means of a *conflict graph*. The vertices of the conflict graph C correspond to the directed edges l_{ij} in the original graph G . There is an edge between vertices l_{ij} and l_{pq} in C if the links l_{ij} and l_{pq} cannot be activated (transmitted on) simultaneously.

In the conflict graph C , we know that vertices belonging to a given *independent set* in C , which represent links in the original connectivity graph G , can be scheduled simultaneously. Therefore an independent set in the conflict graph corresponds to a *schedulable set* of links in the original graph. To avoid interference, the link schedule must ensure that, at any time, all the scheduled links belong to a common schedulable set.

Any link transmission schedule that alternates among schedulable link sets is a feasible one as it does not introduce interference. Therefore, the optimization problem is to find how much of the total time can be spent in each of the schedulable sets, such that the total bandwidth is optimized.

We adapt this generic solution to the specific case of WiLD networks. Here, interference is caused when a node (say j), receives on one link (l_{ij}) while sending on another (l_{jk}). This is equivalent to saying that the conflict graph must have an edge between any two links l_{ij} and l_{jk} of the original graph. Figure 5(a) presents an example of a connectivity graph for a WiLD network, and Figure 5(b) shows its associated conflict graph.

Besides the constraints for *obeying link capacities* and *avoiding interference*, several other constraints can be added in order to reflect the limitations introduced by practical solutions.

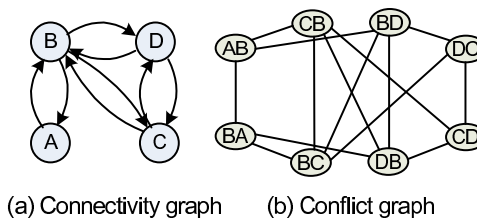


Figure 5: Example of a connectivity graph and its associated conflict graph

One such set of constraints is related to the routing assumptions; if no routing constraints are specified, the *max-flow* solutions assumes multi-path routing. We therefore investigate how the optimal throughput decreases if we constrain the routing to be *single path*, or if we constrain it to be *single path and fixed* to a set of routes computed beforehand (e.g., by using a shortest-path algorithm). We consider these scenarios because most practical routing algorithms make these assumptions.

Finally, we also investigate what happens to the maximum throughput if we constrain the nodes to always transmit simultaneously on all of their links. We use this particular constraint for two reasons: it is assumed by existing approaches such as 2P and WiLDNet, and it also makes the search for schedulable sets much easier and tractable for larger network sizes. Due to space constraints we omit the details of these LP formulations.

Comparison: Practical vs. Optimal: We use the solutions to these LP problems to present the potential for improvement over existing algorithms. We thus compare these solutions against the throughput achieved by existing fixed-slot approaches (FT and FT-CUT). We perform our comparison on the following topologies: a) a 20-node random graph, with an average connectivity degree of 3; b) a real WiLD topology (14 nodes and 19 links) as used in the Aravind Eye Hospital; c) a realistic WiLD topology constructed using the method presented by Raman [17]. We assume a uniform link capacity of 10 Mbps.

To measure saturation throughput (in terms of the maximum number of flows successfully accommodated by the network), we generate an amount of traffic exceeding the maximum capacity. We use CBR flows, with a bandwidth of 500 Kbps. We generate unidirectional flows between random source and destination pairs.

For the offline algorithms, we solve the linear programs generating the throughput-optimal solutions using the ILOG CPLEX [1] optimizer. To evaluate the performance of the online algorithms, we perform simulations using a modified version of the Java-based network simulator developed by Jain [12]. Given that the number of flows accommodated by the network depends on the order in which we add flows, we generate 5 such random flow orderings, and for each ordering we add the flows one by one until we reach saturation. For each run, we find the point when the maximum number of flows was successfully served by the network, and we average among the results obtained in each run. We use this method to compute the throughputs for 5 random topologies for each size, and present the average of these results.

Figure 6 illustrates our comparison. As expected, we find a very large gap between the throughput achieved by practical approaches and the maximum potential throughput. Even with constraints of fixed routing and simultaneous transmission on all the links of a node, the LP solution computed offline outperforms practical solutions by a factor of two. We also see that among the practical algorithms, FT-CUT outperforms FT over the original graph. This happens because, for a graph with a *chromatic number* K larger than 2, sending only once every $K \geq 3$ slots is inefficient.

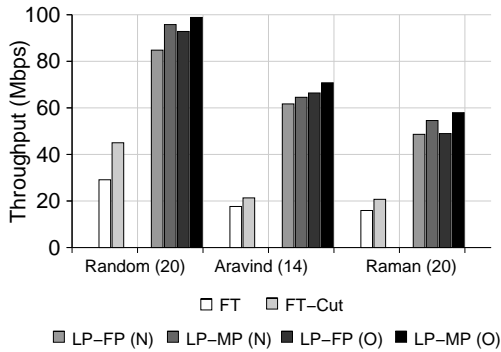


Figure 6: Comparison of maximum throughput for unidirectional CBR flows for the following algorithms: 1)LP-MP (O): LP, multipath; 2)LP-FP (O): LP, fixed path; 3)LP-MP (N): LP, multipath, nodes send to all links simultaneously; 4)LP-FP (N): LP, fixed path, nodes send to all links simultaneously; 5)FT-Cut: FT over maxcut; 6)FT: FT over the original topology

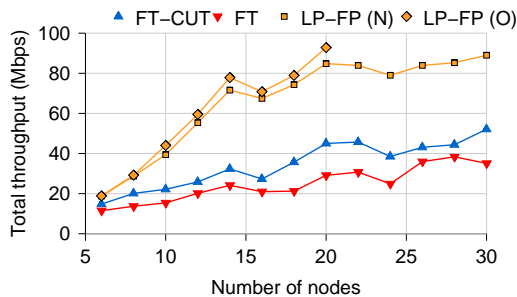


Figure 7: Maximum throughput for unidirectional CBR flows for various protocols with increasing network size. These are random topologies (avg. deg:3).

We also investigate whether this large gap happens in networks of other sizes. Figure 7 compares the network throughput delivered by these algorithms in networks of increasing sizes (# of nodes). For each network size, we generate 5 random topologies with average connectivity degree of 3. Every measurement point corresponds to the average throughput of all the topologies of the same size, each simulated with 5 different random flow orderings as described above. For networks smaller than 20 nodes we compute the fixed-path optimal throughput (LP-FP(O)), while for larger sizes we only compute the approximation where nodes are constrained to transmit simultaneously on all of their links (LP-FP(N)). We find that, at small sizes *e.g.*, 6 nodes), the difference between practical and optimal approaches is small, but this difference increases quickly as we exceed 10 nodes, and remains high afterwards.

Our findings show that existing practical approaches are inefficient over a large spectrum of network topologies, which motivates the development of a new more dynamic MAC layer based on the insights presented above.

4. JazzyMac DESIGN

This section presents JazzyMac, a novel medium access control protocol for long-distance wireless networks that addresses the limitations identified in section 3. Specifically, JazzyMac makes the following key improvements:

Adaptive slots: rather than require fixed-length transmission slots, JazzyMac allows each link to dynamically adapt the length of its transmission slots based on locally observed traffic load. Adaptive slots lead to more efficient bandwidth allocations and also allows greater flexibility in navigating the tradeoff between throughput

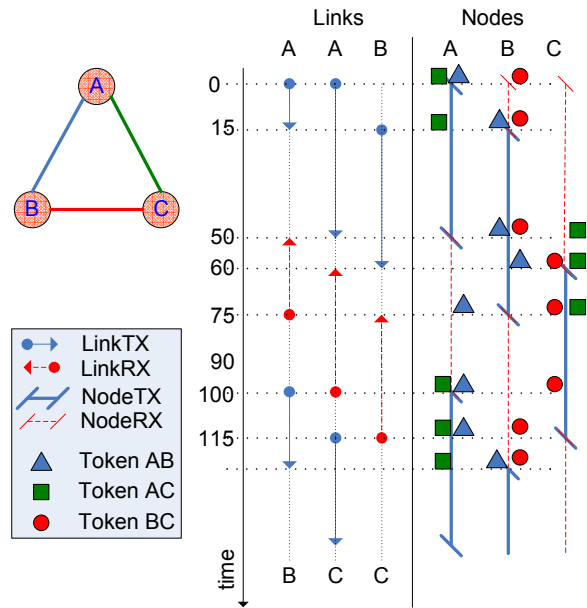


Figure 8: Scenario featuring three nodes and three links. The figure presents the network topology, and illustrates how data is sent and received on each of the network links. The figure also shows how nodes transition between TX and RX states, as well as the distribution of the link tokens between the three nodes.

and delay.

Allow parallel neighboring-but-independent transmissions: the protocol is specifically designed to allow neighbors to proceed with parallel independent transmissions, as exemplified in section 3, which contributes to increased throughput.

Generalized topologies: scheduling in JazzyMac does not require that the topology be bipartite, making the protocol applicable to arbitrary topologies.

JazzyMac achieves the above using simple and fully distributed algorithms that rely only on readily available local state. This makes JazzyMac practical for implementation in existing radios and hardware platforms.

4.1 Protocol Description

We now describe the JazzyMac protocol. Every node A is associated with a node-wide **mode** of operation, which can be either transmit (TX) or receive (RX). Each network link AB is associated with a **token**, T_{AB} , that is at all times in the possession of either node A or B and only the node holding the token can transmit on the associated link. In addition, each token is associated with a **timeout value**, v_{AB} , that controls *when* the node holding the token is allowed to transmit over the associated link. Finally, we introduce a network-wide parameter `max_slot` that bounds the maximum length of any transmission slot.

Given the above protocol state, the basic operation of JazzyMac is guided by the following four rules:

(1) token exchange rule: When a node (say) B has completed its transmission over link AB , it computes a timeout value v_{AB} that estimates the time in the future when node B will be willing to receive traffic *from* A (we describe how v_{AB} is computed shortly). Node B then hands the tuple (T_{AB}, v_{AB}) to node A . If node A receives this token at time t , we say that token T_{AB} is *valid* after time $t + v_{AB}$.

(2) mode rule: A node B that is in receive mode can transition to transmit mode only when it holds the token (whether valid or not)

for *all* its links. Likewise, a node returns to receive mode when it has released the tokens for *all* its links.

(3) transmission rule: A node A can transmit over link AB only when the following two conditions are true: (1) node A is in transmit mode and (2) node A holds token T_{AB} , and T_{AB} is valid. (Note that, by the mode rule, A being in transmit mode ensures it has T_{AB}).

(4) slot rule: A node A can transmit on link AB for no longer than max_slot time units.

Figure 8 illustrates the operation of JazzyMac for a simple 3 node scenario. Assume that node A initially holds the tokens for links AB (T_{AB}) and AC (T_{AC}), while node B holds the token T_{BC} . The timeline proceeds as follows.

1. At $t = 0$, since node A has all the tokens, it is in node-wide TX mode and starts transmitting on both its links.
2. At $t = 15$, A 's transmission to B ends, and token T_{AB} is passed to B . Note that A 's transmission to C lasts much longer (50 time units). Therefore, T_{AB} is passed with a timeout $v_{AB} = 35$, the additional time until node A finishes its transmission to C .
3. Also at $t = 15$, node B has all its tokens and hence transitions into a node-wide TX mode. However only token T_{BC} is valid, and therefore B starts transmitting only to node C . In prior MACs, to avoid collisions, B would transmit to C only when A finished *all* its transmissions. With JazzyMac, we can permit such *neighboring-but-independent* transmissions without resulting in any collisions.
4. At $t = 50$, A releases token T_{AC} and transitions into node-wide RX mode. B 's token T_{AB} becomes valid and it starts transmitting over link AB .
5. At $t = 60$, C transitions to TX, and so on.

Note that the use of a node-wide mode of operation controlled by the above rules ensures that JazzyMac respects the fundamental limitation of inter-link interference in WiLD networks. Specifically, node A never transmits on link AB while receiving on another link (say) CA and also never transmits on link AB while node B is itself transmitting on some link BC .

The use of token timeouts v_{AB} allows neighboring nodes to simultaneously transmit provided these transmissions are independent. For example, in the above scenario, nodes A and B can simultaneously transmit between times 15 and 50. This allows JazzyMac to move beyond the strict alternation imposed by solutions based on bipartite scheduling. In addition, we show in Section 4.3 that the above rules suffice to ensure that JazzyMac is deadlock and starvation free.

We now address two additional questions not addressed by the above protocol description.

(#1) How long does a link transmission last? The max_slot parameter sets the upper limit on slot lengths. To select a good slot length, JazzyMac selects a slot length based on its locally observed traffic demand. Our implementation uses the *per-link outgoing queue length* as a measure of traffic demand on the link in question. Let tt_{AB} denote the estimated time to transmit all the packets queued for transmission over link AB . The slot length for link AB is then selected to be the minimum of tt_{AB} and max_slot . This policy allows busy links to transmit for longer, and less used links to transmit for shorter periods, as demanded by network traffic. The max_slot bound ensures fairness, in terms of a minimum per-link bandwidth and packet delay (to be discussed in section 4.3).

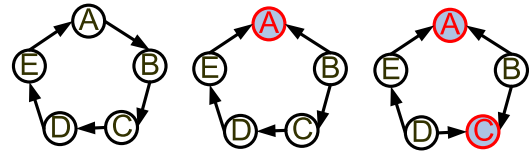


Figure 9: Example initial token assignments

(#2) How are timeout values v_{AB} calculated? As described above, when node A finishes its transmission on link AB , it must calculate a timeout period v_{AB} that estimates the time when node A exits transmit state and is ready to receive traffic from B . The difficulty is that in order to estimate v_{AB} , node A must estimate the time *in the future* when it will be done transmitting on *all* its links. We implement this by estimating a remaining-transmission time for each link individually and setting v_{AB} to the maximum of these estimates. For the links that are done transmitting, the estimated pending transmission time is zero; for links that are already transmitting, the overall slot time is already known (calculated using #1 above) and hence the remaining transmission time is known. For links over which transmission hasn't yet begun (*e.g.*, if the token for the link is still inactive), we estimate the remaining transmission time as the sum of the time left to the activation of the link token and the time required to transmit packets currently buffered at the link's outgoing queue. After estimating all the per-link transmissions end times, the latest of these times is selected as v_{AB} , and subsequently advertised to peers when exchanging tokens. Once the end of the node-wide TX has been established, all links will make sure not to transmit past this time.

The above completes the description of the basic JazzyMac protocol operation. In addition, we must specify *a)* how is the protocol bootstrapped (in terms of the initial token assignment), and *b)* how does JazzyMac recover from token losses and node failures. We describe our bootstrapping protocol in the following section, and discuss recovery mechanisms in Section 4.4.

4.2 Protocol Bootstrapping

The protocol liveness and efficiency depend on the initial assignment of link tokens. While the long-term functioning of our protocol is distributed and requires only local information, the initial token assignment will be computed globally during the network planning phase (in future work we plan to investigate distributed coloring in dynamically-changing networks).

In order to illustrate the effect of different initial assignments, let us examine different possibilities for the 5-node cycle presented in figure 9. Assigning a link token is similar to establishing an initial direction for the given link. In this example, we make the simplifying assumption that all transmission slots have the same length. Some possible initial states are:

- If we decide to start by giving one token to each node, the protocol will be in a deadlock situation, since none of the nodes can proceed with their transmissions (figure 9(a)).
- If we begin in the state in which only one node (node A) has all its tokens (figure 9(b)), then node A sends first, followed by B , then by C , then D , then E and finally A again, *i.e.*, one node at a time. Thus, each link transmits for 20% of the time in each direction, and is idle for 60% of the time (please be reminded that our example assumes equal-sized transmit slots).
- If we begin with the initial assignment presented in figure 9(c), where at the beginning both nodes A and C can immediately (and simultaneously) start transmitting, then nodes

A and C go together, followed by the pair B and E , then by A and D , then C and E , and so on. The sets of nodes that send at one time keep changing, with 2 nodes always transmitting simultaneously. In this scenario (which is the optimum one for fixed slots), links send for 40% of the time in each direction, and are idle for 20% of the time.

Thus we see that the steady-state performance of JazzyMac is determined by the initial protocol state. We therefore aim to assign an initial state that allows JazzyMac to ensure and maintain the following correctness and performance-related properties:

- a) deadlock-free operation
- b) starvation-free operation (every node gets the opportunity to send),
- c) a lower bound on the fraction of time in which a link can send in each direction (provided that the link requires this much time for transmission), and
- d) an upper bound on the per-link packet delay time.

We therefore propose the following bootstrapping algorithm:

1. Color the vertices of the network graph with the minimum number of colors K such that no two adjacent vertices have the same color.
2. The tokens are assigned to the link end that has the lowest color (the two ends must be colored differently).

4.3 JazzyMac Properties

In the following we prove that JazzyMac is deadlock-free (assuming the bootstrapping strategy introduced earlier), and that it observes a set of performance guarantees in terms of *link utilization* and *per-hop maximum packet delay*. In the interest of space, we only give the formal proof for the simpler case assuming fixed slot sizes, and provide the intuition for why the same properties hold for the general case of dynamic slot sizes.

Fixed slot size case: In this simplified case, time can be regarded as a succession of equally-sized time slots. For our proof, we introduce the following abstraction that describes the protocol in a manner equivalent to our token-based description. Imagine that each node has a non-decreasing sequence number. Let $S_i(A)$ be the sequence number of node A in slot i , and let $T_i \subseteq G$ be the set of nodes transmitting during slot i . In the initial slot, the sequence number of every node is equal to the vertex color used to bootstrap the algorithm: $S_0(A) = \text{color}(A)$. After nodes transmitting in slot i finish sending, they recompute the value of their sequence number to be one larger than the maximum sequence number of their neighbors:

$$S_{i+1}(A) = 1 + \max_{X \in \text{neighbors}(A)} S_i(X), \forall A \in T_i$$

while the non-transmitting nodes $B \notin T_i$ keep their sequence numbers unchanged: $S_{i+1}(B) = S_i(B)$.

Using these sequence numbers, the condition to be fulfilled by node A in order for A to belong to the set T_i (meaning that A has the tokens for all its links) can be expressed as:

$$A \in T_i \iff S_i(A) < S_i(N), \forall N \in \text{neighbors}(A) \quad (2)$$

. We continue by stating the following property.

PROPERTY 1. *During any time slot, the difference in sequence numbers between any two network nodes remains strictly smaller than the number of colors used for graph coloring:*

$$\max_{A \in G} S_i(A) - \min_{B \in G} S_i(B) \leq \text{colors}(G) - 1 \quad (3)$$

PROOF. By induction. We use the initial assignment of sequence numbers as the base case, and for this base case Property 1 holds, because all nodes have sequence numbers between 1 and the maximum number of colors. For our inductive step, we assume that the property holds in slot n , and we prove it for slot $n + 1$. Since in slot n there is at least one node that has a sequence number smaller than the ones of its neighbors, $T_n \neq \{\}$. Also, the set M of nodes that have the *minimum* sequence number in the entire network is a subset of T_n . During slot n , every node $A \in T_n$ transmits and then sets its sequence number to $1 + \max_{N \in \text{neighbors}(A)} S_n(A)$. Since $M \subset T_n$, and all the sequence numbers of nodes $A \in T_n$ increase by at least one, it means that in slot $n + 1$:

$$\min_{B \in G} S_{n+1}(B) \geq \min_{B \in G} S_n(B) + 1 \quad (4)$$

On the other hand, $\max_{N \in \text{neighbors}(A)} S_n(N) \leq \max_{P \in G} S_n(P)$, and therefore the maximum sequence number in the network will not increase by more than one:

$$\max_{B \in G} S_{n+1}(B) \leq \max_{B \in G} S_n(B) + 1 \quad (5)$$

From (3,4,5) it follows that:

$$\max_{A \in G} S_{n+1}(A) - \min_{B \in G} S_{n+1}(B) \leq \text{colors}(G) - 1 \quad (6)$$

which concludes our proof. \square

PROPERTY 2. *There protocol does not result in any deadlock or node starvation.*

PROOF. Knowing that property (1) holds, it becomes obvious to show that there is no starvation. This follows from the fact that, at every slot, the minimum sequence number in the network increases by at least one (as previously shown) and therefore in any K consecutive slots, the minimum sequence number increases with at least K . But since, at any time, all the nodes have sequence numbers that differ by at most K , we can conclude that every node must transmit at least once every K slots. Therefore *none of the nodes will starve* (this obviously implies that there is also no *deadlock*). \square

The proof above directly entails the following properties:

PROPERTY 3. *Every node can choose to send on each of its links for at least $1/K$ of the link capacity.*

PROPERTY 4. *The maximum delay between two consecutive opportunities to send on any link is smaller than $1/K$*

These two properties establish performance guarantees, the former introducing a lower bound on link utilization, and the latter introducing an upper bound on per-link delay.

Dynamic slot sizes: The properties above also hold for the general JazzyMac protocol, and we provide a brief intuition for it here. The first observation we make is that a node using variable slots goes through the same sequence of node-wide TX and RX states as when using fixed slots. Furthermore, the token exchanges performed by a node during a particular TX or RX state is also same as in the fixed slot case. The difference between the two scenarios is given by the fact that, with variable slots, nodes have the option to give up tokens *earlier* than in the fixed case. These observations can be used to show that a particular token exchange in the variable slot case can only happen earlier than the same exchange in the fixed slot case. Therefore, at any time from the beginning of operation, each link would have had at least as many opportunities to transmit as in the fixed slot size case. This means that the protocol does not suffer from starvation, and obeys similar bandwidth bounds.

4.4 Dealing with Loss

In the previous section we presented the behaviour of the JazzyMac protocol under ideal conditions, in which packet losses, link and node failures do not happen. In this section, we briefly discuss how can JazzyMac be provisioned to deal with some of these possible occurrences.

Even though JazzyMac eliminates interference at co-located radios, other sources of packet loss such as *external interference* can still cause packet loss in long distance links [21, 5]. This can lead to *loss of link tokens*, affecting the functioning of our protocol.

Consequently, any JazzyMac implementation should take precautionary measures in order to minimize the probability of losing tokens. There are several ways to make the protocol more resilient to such occurrences, including piggybacking tokens on several data packets, and sending multiple copies of the token and validation in small packets.

However, in the unlikely event that the loss still occurs, our protocol must recover properly. This is a delicate issue: simply assuming token receipt after waiting for a certain *timeout* period is not adequate, because it breaks the inter-node ordering established during bootstrapping – possibly leading to starvation or low-performance steady-state operation. For example, consider the chain A-B-C-D, and assume the loss of token T_{AB} . This will prompt B to wait, which in turn would prompt C to wait for token T_{BC} . Now if node C assumes to have received token T_{BC} after a timeout, we arrive into a situation where both nodes B and C believe that they hold token T_{BC} . Moreover, if node C goes ahead to transmit to its neighbors, the ordering between nodes is broken. In order to maintain the original inter-node ordering, we must make sure that the lost tokens are recovered, while the rest of the token exchanges remain unaffected.

We propose a solution that involves adding a sequence number S_{AB} to each token T_{AB} – set to 0 during bootstrapping. At every valid exchange of T_{AB} , S_{AB} is incremented. The solution works as follows: If a token T_{AB} is lost, the recipient (B) will wait for it, which will prompt other nodes, including A , to wait as well. After a *timeout* given by the maximum time between successive link transmissions ($K \times \text{max_slot}$), every node will resend the tokens they have sent last. Duplicate tokens (that have previously been received) will be ignored, and the lost token (resent by A) will be properly recovered.

A problem with this approach is that simultaneous token retransmissions by several hosts can interfere with each other or other packets. To minimize the probability of such occurrences, the tokens can be sent in small packets, at random intervals after the timeout and retransmitted periodically until successful.

The same sequence numbers can be used to detect a link (or node) that is permanently down: if a node A does not receive any retransmitted tokens on a quiescent link AB for a long time, the link is marked *down*. From that moment on, A will not wait for tokens on the link AB . Instead, it will transmit a copy of the token during every TX state, and assume its instantaneous return. In the event that a node dies, all its links will be individually marked as being down.

When a node B re-joins the network after a period of inactivity, it first listens on all of its links (to verify that they are still active), and then advertises its presence by sending special *join request* packets on all its active links. These requests are repeated periodically, until a response is received. Upon receiving a join request, the neighboring nodes will mark the respective link active again, and respond by sending the link token to B . Upon receiving all the link tokens, B will resume normal operation. As in the case of token retransmission, join requests can create an acceptable amount of interference.

5. EVALUATION

In this section we evaluate the performance of JazzyMac over a range of topology types and sizes, and various patterns of traffic. Our findings show that:

- JazzyMac greatly improves the maximum network throughput achievable by existing approaches: 15–100% in typical topologies, depending on traffic.
- The throughput improvements are consistent across many topologies and traffic patterns. Improvements are highest in asymmetric traffic patterns, when dynamically sized slots can offer much better bandwidth allocation compared to fixed-slot approaches.
- Our protocol significantly reduces the gap between network throughput achieved by practical approaches and the optimal network throughput computed offline, and assuming idealized transmission slots (no switching overhead).
- Dynamic slot sizing improves the delay-throughput tradeoff, offering increased maximum throughput when required, and decreased packet delay at average network utilizations.
- Having the ability to operate over non-bipartite topology is important, allowing throughput increases as large as 80% with asymmetric traffic distributions.

5.1 Methodology

In our experiments we compare the performance of fixed TDMA approaches with JazzyMac when run on both the original network connectivity graph (FT and JZ) and the bipartite subgraph (maxcut) (FT-CUT and JZ-CUT):

	Fixed Slots	Adaptive Slots
Original Graph	Fixed TDMA (FT)	JazzyMac (JZ)
Maxcut	Fixed TDMA on cut (FT-CUT)	JazzyMac on cut (JZ-CUT)

We run our experiments using a version of the Java-based network simulator developed by Jain [12], modified with MAC-level support for our protocols. We consider a range of network topologies and sizes, and various traffic demand patterns. We assume a link capacity of 10 Mbps. The link propagation delay is not considered explicitly, but is accounted for in the slot switch time (guard time) t_{switch} , which we conservatively set to 1ms in the default case. Unless otherwise specified, we use a slot size (or maximum slot size for adaptive algorithms) of 20ms. We assume no packet or token losses, and no node or link failures. For the optimal LP formulations we use the experimental setup described in Section 3.4.

Topologies: We consider several types of topologies: a) random topologies, with varying degrees of connectivity and of varying sizes; b) an actual real-world topology, derived from the one used in the Aravind Eye Hospital in India [23]; c) typical mesh WiFi topologies, using the construction method introduced by Raman [17], which we denote as the *Raman* topology henceforth.

Traffic: We assume traffic consisting of many CBR flows, 500 Kbps each. We choose CBR flows because this traffic is representative of applications supported today in rural wireless deployments: VoIP, telemedicine and streaming of educational content.

We consider the following patterns of traffic demand, ranging from very *asymmetric* to very *symmetric*: a) one source to many randomly distributed destinations (*single-source, many sinks*); b) unidirectional CBR flows, with randomly chosen source-destination pairs (*unidirectional*); c) pairs of CBR flows in opposite directions (*bidirectional*), with many random source-destination pairs. We leave the evaluation of other types of traffic (e.g., TCP) for future work.

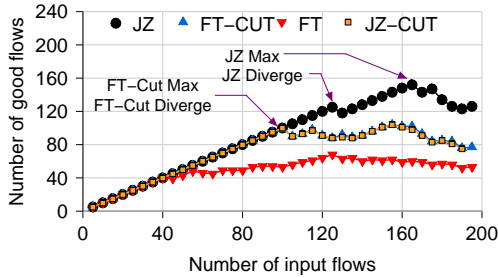


Figure 10: Number of good flows as we add *unidirectional* CBR flows on a 30-node random graph (avg. deg:3)

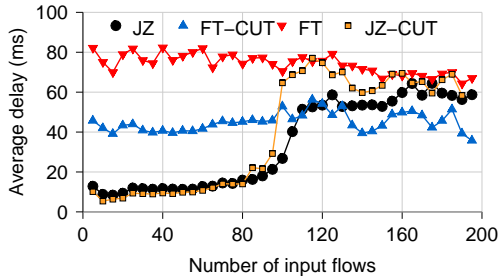


Figure 11: Average delay of good flows as we add *unidirectional* CBR flows on a 30-node random graph (avg. deg:3)

Performance metrics: We measure performance in terms of maximum network throughput and average delay.

Method: We generate several networks (typically 5) for each topology type and network size. Next, we generate several ordered sets (typically 5) of CBR flows, using the traffic patterns described above. We then run a simulation using each of these flow sets. During a simulation run, we start with an unused network, and incrementally add CBR flows from the ordered flow set, until the network reaches saturation.

5.2 Performance in Random Topologies

In order to understand the behaviour of various algorithms with increasing network load, we start our evaluation by looking at one graph, a 30-node random graph with an average degree of 3, and we use randomly generated *unidirectional* CBR traffic.

We perform the experiment by adding one CBR flow at a time; we then measure how many of these flows can be accommodated by the network. We consider a flow to be accommodated successfully (*i.e.*, a “good” flow) if it receives 90% of its packets and its per-packet delay is not continuously increasing.

As we can see from our results, plotted in Figure 10, initially all flows are accommodated, but as more flows are added, some links become saturated, and the corresponding flows on these saturated links suffer. For each experiment, we emphasize two metrics: a) *max point*, which is the maximum number of good flows supported at any time during the experiment (we use this as a proxy for maximum throughput), and b) *divergence point*, which is the maximum number of good flows that could be successively added from the beginning without having to drop any flow. We highlight this second metric because in practice, flows arrive in random order and, once accepted, they cannot be dropped. These two metrics are illustrated in Figure 10. For this particular network, JazzyMac accommodates 46% more flows than FT-CUT and 137% more than FT. JazzyMac is also 32% better at the *divergence point* than FT-CUT and over 200% better than FT.

Figure 11 shows what happens to the average per-flow delay

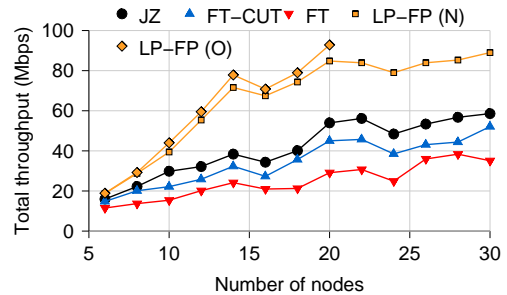


Figure 12: Maximum throughput for *unidirectional* CBR flows with increasing network size. LP-FP(O) is the fixed path optimum (not tractable beyond size 20), and LP-FP(N) is the more tractable approximation (described in section 3.4). For random topologies (avg. deg:3)

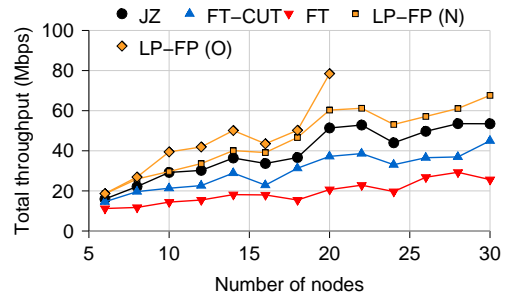


Figure 13: Divergence throughput or *unidirectional* CBR flows with increasing network size. For random topologies (avg. deg:3)

as we increase the volume of traffic in the network. As expected, JazzyMac operates at lower slot sizes, and therefore sees consistently smaller delays than FT and FT-CUT, which use a fixed slot size of 20ms. At low utilizations, these delay improvements are between 2–4x over FT-CUT and between 4–8x over FT.

Next we investigate whether JazzyMac’s improvements are consistent over a range of network sizes. We start by revisiting the experiment discussed in Section 3, where we measure the gap in maximum throughput between practical and optimal algorithms. For this experiment we span several network sizes, and compute average results over 5 random topologies for each network size, and 5 sets of unidirectional CBR traffic demands for each topology. Our results, presented in Figure 12 confirm that JazzyMac outperforms fixed slot approaches in all cases, and that it reduces the gap to optimal throughput.

Nonetheless, this gap remains large. Upon closer inspection, we find this to be the case because the solution of the LP chooses the best set of flows (that maximizes total throughput) from the large pool of input flows. In contrast, JazzyMac and the other practical algorithms are constrained to accept flows in the order they arrive (which is non-optimal), making our comparison unfair. To compensate for this effect, we compute the equivalent of the *divergence point* for the LP solution, using the following iterative approach: we start with a small set of flows, and incrementally add flows to it. At each step, we run the LP solver to see if the current flow set is *feasible*, meaning that the LP solver can find a link transmission schedule that accommodates all of the flows in the set. If this is the case, we add more flows, otherwise we stop, and use the network throughput achieved using the largest *feasible* flow set as our divergence point. We then compare the *divergence throughput* of the optimal approaches to the *divergence throughput* of JazzyMac and

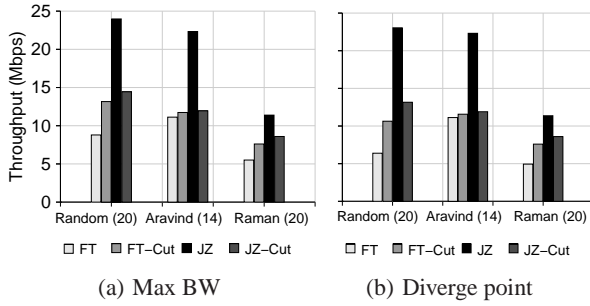


Figure 14: Throughput for various topologies, random CBR flows from one source to all the nodes.

other practical algorithms, and plot the results in Figure 13. We can see that, after eliminating the unfairness in our comparison, JazzyMac effectively halves the gap to optimal throughput.

5.3 Effect of Traffic and Topology

We now examine the performance of JazzyMac under various traffic patterns and various topology types.

One source, many sinks: In our first experiment we use traffic from a single source to many sinks, which is representative of the case when a video server streams content to many destinations. Figure 14 plots the (a) *maximum bandwidth* and (b) the number of flow additions until the first network flow fails (*i.e.*, the *divergence point*), achieved by our algorithms in the three types of topologies described previously (random, Aravind, and Raman). The random and Raman topologies have 20 nodes, while the Aravind topology has 14 nodes. For the random and Raman cases, we generate 5 topologies and for each topology we run 5 sets of CBR flows and average over all of them. For the Aravind topology also, we average over 5 sets of flows.

We find that for this asymmetric traffic distribution, JazzyMac achieves dramatically higher throughput across all topologies, with improvements as large as 100% over FT-CUT, and even larger over FT. We also note that, while FT performs better over the maxcut (*i.e.*, $FT-CUT > FT$), JazzyMac performs much better over the original graph: it is able to make productive use of the extra links.

Many sources and sinks: We perform the same comparisons for the other two traffic patterns as well: random unidirectional (Figure 15) and bidirectional (Figure 16) flows. In the first case, the improvements over FT-CUT are between 25–50% for the *max point*, and 40–60% for the *divergence point*. In the second case, the throughput improvements are much lower, 15–45% for the *max point* and 20–50% for the *divergence point*.

Overall, we see that JazzyMac consistently outperforms the other protocols across all the topologies and traffic types. We also find that the relative throughput improvements given by JazzyMac are larger for more asymmetric traffic. This is to be expected, given that variable slot sizes are most useful in asymmetric traffic conditions, where traffic demands are very different in different directions on the same link, but also across different links. For symmetric traffic, which naturally requires similar slot sizes, JazzyMac’s throughput improvements are more modest.

Another important observation can be derived from the relative ordering (in terms of achieved throughput) of the four measured protocols: $JZ > JZ-CUT \approx FT-CUT > FT$, which holds true across all our topologies and traffic patterns. On one hand, this finding confirms that, for fixed slot approaches, it is more opportune to operate on the maximum bipartite subgraph of a given network topology (as done by 2P and WiLDNet) rather than on the original topol-

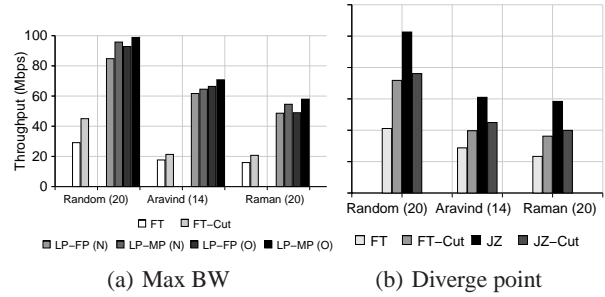


Figure 15: Throughput with *unidirectional* random CBR flows.

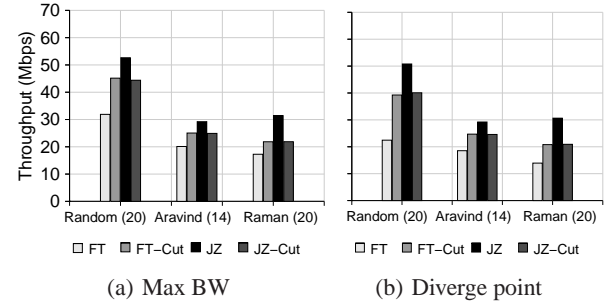


Figure 16: Throughput with *bidirectional* random CBR flows.

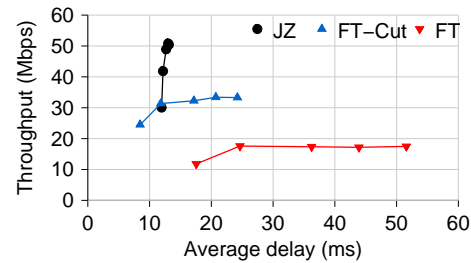


Figure 17: Maximum throughput vs. average delay.

ogy (since $FT-CUT > FT$). On the other hand, if dynamic slots are used, it becomes more profitable to operate on the original non-bipartite topology ($JZ > JZ-CUT$). This confirms the importance of having an approach that takes advantage of all the network links, increasing network capacity (reflected in the increased throughput achieved by JZ), but also improving fault-tolerance.

5.4 Bandwidth vs. Delay Tradeoff

We also look at how JazzyMac enables a better combination of *average delay* and *maximum throughput* than existing fixed-slot approaches. We perform our experiments using a random topology of 20 nodes with an average connectivity degree of 3, and using random unidirectional CBR traffic. To change the tradeoff between throughput and delay, we vary the *fixed* slot sizes of FT-CUT and FT uniformly, between 3ms and 12ms, in 5 steps. For JazzyMac, we vary the value of the *maximum* slot size in the same range. For all of the algorithms, we assume a slot switching overhead of 1ms.

We then measure the *maximum network throughput*, and the *average end-to-end delay* experienced by the network at a throughput level equal to half of the saturation throughput of the network. We plot the tradeoff between maximum throughput and average delay in Figure 17.

We first note that setting JazzyMac’s upper bound on the slot size to the largest value (12ms) is clearly the best setting in terms of throughput, and as good as any other setting in terms of delay. This result confirms that JazzyMac’s adaptation mechanism is effective

in dynamically adapting slot size to achieve both high throughput when needed and low delay at average utilizations.

We also find that, as suggested by previous experiments, Jazzy-Mac outperforms FT-CUT and FT by a large margin, in terms of both throughput and delay. Among the fixed-slot approaches FT-CUT performs best, and increasing its fixed slot size beyond $6ms$ has diminishing bandwidth benefits.

6. OTHER RELATED WORK

WiLD Networks: The use of 802.11 has grown beyond its originally intended purpose of indoor wireless LANs to multi-hop outdoor meshes, both short range [3] and long range [16, 19]. It is a well known fact that CSMA/CA is ill-suited for multi-hop mesh network settings (short and long-range) [9, 13, 21]. As a result, several TDMA-based protocols have been proposed for mesh networks.

Maximizing throughput in multihop wireless networks: Djukic and Valaee [7] propose min-max heuristics to provide offline algorithms to minimize delay when link bandwidths *are known in advance*. Wang et al. [25] provide centralized and distributed algorithms to maximize throughput by taking into account interfering links. In comparison, our new protocol needs no future knowledge of traffic, is fully distributed, provides flexible delay-bandwidth guarantees, and can dynamically adjust to varying traffic demands.

MAC implementations using 802.11 radios: Several MAC implementations using 802.11 radios have been proposed. Most related to ours are 2P [19] and WiLDNet [16] (covered earlier). Overlay-MAC [20] provides a deployable approach towards implementing a TDMA-style MAC on top of 802.11 MAC hardware. Softmac [15] is a platform that can be used to build experimental MAC protocols. MultiMAC [8] extends this approach so that multiple MAC layers can co-exist and any one can be chosen on a per-packet basis. These approaches are complementary to our work as we can build JazzyMac over these platforms.

7. CONCLUSION

WiLD networks provide network access, VoIP and telemedicine services to many thousands of users in rural areas around the world. These networks use standard WiFi radios and TDMA-based MACs to achieve good throughput over multi-hop long distance networks. Although these approaches provide real gains over CSMA-based solutions, they are limited by their use of fixed-sized TDMA transmission slots, and constrained by interference to operate only over bipartite network topologies. In this work we have identified opportunities, based primarily on dynamic slot sizing according to traffic demand, to further improve throughput, to reduce latency, and to enable operation on general topologies.

We therefore present JazzyMac, a fully distributed, practical MAC layer that uses local traffic information to adapt link transmission slot sizes dynamically. JazzyMac uses dynamic slot sizing to negotiate the delay-throughput tradeoff in WiLD networks, and exploits asymmetric traffic, time varying traffic, and non-bipartite topologies to achieve a much higher throughput than existing TDMA-based approaches. Our protocol consistently outperforms existing WiLD MAC protocols in terms of throughput and average latency, and can operate unconstrained in any network topology.

8. REFERENCES

- [1] CPLEX: LP Solver. <http://www.ilog.com>.
- [2] P. Bhagwat, B. Raman, and D. Sanghi. Turning 802.11 Inside-out. In *HotNets-II*, 2004.
- [3] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and Evaluation of an Unplanned 802.11b Mesh Network. In *ACM MOBICOM*, Aug. 2005.
- [4] K. Chebrolu and B. Raman. FRACTEL: A Fresh Perspective on (Rural) Mesh Networks. In *ACM SIGCOMM Workshop on Networked Systems for Developing Regions (NSDR)*, August 2007.
- [5] K. Chebrolu, B. Raman, and S. Sen. Long-Distance 802.11b Links: Performance Measurements and Experience. In *ACM MOBICOM*, 2006.
- [6] Connecting Rural Communities with WiFi. <http://www.crc.net.nz>.
- [7] P. Djukic and S. Valaee. Link Scheduling for Minimum Delay in Spatial Re-use TDMA. In *IEEE INFOCOM*, May 2007.
- [8] C. Doerr, M. Neufeld, J. Filfield, T. Weingart, D. C. Sicker, and D. Grunwald. MultiMAC - An Adaptive MAC Framework for Dynamic Radio Networking. In *IEEE DySPAN*, Nov. 2005.
- [9] V. Gambiroza, B. Sadeghi, and E. Knightly. End-to-End Performance and Fairness in Multihop Wireless Backhaul Networks. In *ACM MOBICOM*, 2004.
- [10] HyperLink HG2424G 2.4GHz 24dBi High Performance Die Cast Reflector Grid Wireless LAN Antenna. http://www.hyperlinktech.com/multimedia/datasheets/DS_HG2424G-NF.PDF.
- [11] K. Jain, J. Padhye, V. Padmanabhan, and L. Qiu. Impact of Interference on Multi-hop Wireless Network Performance. In *ACM MOBICOM*, Sept. 2003.
- [12] S. Jain, K. Fall, and R. Patra. Routing in a Delay Tolerant Network. In *ACM SIGCOMM*, 2004.
- [13] K. Jamieson, B. Hull, A. K. Miu, and H. Balakrishnan. Understanding the Real-World Performance of Carrier Sense. In *ACM SIGCOMM E-WIND Workshop*, Aug. 2005.
- [14] M. Kodialam and T. Nandagopal. Characterizing the Achievable Rates in Multihop Networks. In *ACM MOBICOM*, Sept. 2003.
- [15] M. Neufeld, J. Fifield, C. Doerr, A. Sheth, and D. Grunwald. SoftMAC - Flexible Wireless Research Platform. In *HotNets-IV*, Nov. 2005.
- [16] R. Patra, S. Nedeveschi, S. Surana, A. Sheth, L. Subramanian, and E. Brewer. WiLDNet: Design and Implementation of High Performance WiFi Based Long Distance Networks. *USENIX NSDI*, 2007.
- [17] B. Raman. Channel Allocation in 802.11-based Mesh Networks. In *IEEE INFOCOM*, Apr. 2006.
- [18] B. Raman and K. Chebrolu. Revisiting MAC Design for an 802.11-based Mesh Network. In *HotNets-III*, 2004.
- [19] B. Raman and K. Chebrolu. Design and Evaluation of a new MAC Protocol for Long-Distance 802.11 Mesh Networks. In *ACM MOBICOM*, Aug. 2005.
- [20] A. Rao and I. Stoica. An Overlay MAC layer for 802.11 Networks. In *MOBISYS*, June 2005.
- [21] A. Sheth, S. Nedeveschi, R. Patra, S. Surana, L. Subramanian, and E. Brewer. Packet Loss Characterization in WiFi-based Long Distance Networks. In *IEEE INFOCOM*, 2007.
- [22] L. Subramanian, S. Surana, R. Patra, M. Ho, A. Sheth, and E. Brewer. Rethinking Wireless for the Developing World. In *Hotnets-V*, 2006.
- [23] S. Surana, R. Patra, S. Nedeveschi, M. Ramos, L. Subramanian, Y. Ben-David, and E. Brewer. Beyond Pilots: Keeping Rural Wireless Networks Alive. In *USENIX NSDI*, 2008.
- [24] The Akshaya E-Literacy Project. <http://www.akshaya.net>.
- [25] W. Wang, Y. Wang, X.-Y. Li, W.-Z. Song, and O. Frieder. Efficient Interference-Aware TDMA Link Scheduling for Static Wireless Networks. In *ACM MOBICOM*, 2006.